

Zadanie szkoleniowe – Jeden spójny program (Python)

Celem zadania jest zaprojektowanie i zaimplementowanie jednego, spójnego programu w Pythonie, który łączy w sobie obsługę plików, klasy abstrakcyjne, protokoły, asocjację oraz agregację. Zadanie kładzie nacisk na architekturę i świadome użycie relacji obiektowych, a nie wyłącznie na poprawne działanie kodu.

Kontekst programu

Projektowany program ma realizować prosty system przetwarzania danych z plików. Program czyta dane z plików wejściowych, przetwarza je zgodnie z ustalonym scenariuszem, a następnie zapisuje wynik do pliku wyjściowego. Dane mogą reprezentować np. listę transakcji, pomiarów, użytkowników lub produktów (konkretny temat danych pozostawiony jest do wyboru kursanta).

Zakres funkcjonalny programu

Program powinien:

- wczytywać dane z plików tekstowych (TXT) oraz plików CSV,
- przetwarzać dane w ujednolicony sposób (np. filtrowanie, sumowanie, statystyki),
- zapisywać wynik przetwarzania do pliku wyjściowego,
- umożliwiać łatwe rozszerzenie o nowe formaty plików lub nowe sposoby przetwarzania.

Obsługa plików

Zaprojektuj część programu odpowiedzialną za obsługę plików:

- bezpieczne otwieranie i zamykanie plików,
- obsługę błędów (np. brak pliku, błędny format),
- przetwarzanie danych w sposób niewymagający wczytywania całego pliku do pamięci.

Program powinien czytelnie oddzielać logikę pracy z plikami od logiki biznesowej.

Klasy abstrakcyjne (ABC)

W projekcie należy użyć klasy abstrakcyjnej, która opisuje wspólne zachowanie dla elementów systemu (np. źródła danych lub procesora danych).

Klasa abstrakcyjna powinna:

- definiować zestaw metod wymaganych od klas potomnych,
- uniemożliwiać użycie niekompletnej implementacji,
- stanowić stabilny punkt rozszerzeń programu.

Protokoły (Protocol)

W programie należy użyć co najmniej jednego protokołu (Protocol), który opisuje wymagane zachowanie obiektów bez narzucania dziedziczenia.

Protokół powinien:

- definiować zestaw metod wymaganych do współpracy z innymi elementami systemu,
- być wykorzystywany jako typ zależności (np. w konstruktorach lub funkcjach),
- umożliwiać łatwe podstawianie implementacji testowych lub alternatywnych.

Asocjacja

Zastosuj relację asocjacji pomiędzy obiektami w systemie.

Przykładowo:

- główny obiekt zarządzający przetwarzaniem danych przechowuje referencję do obiektu odpowiedzialnego za zapis wyników,
- obiekt raportu posiada referencję do obiektu źródła danych.

Relacja asocjacji powinna oznaczać współpracę obiektów, bez przejmowania kontroli nad cyklem życia któregoś z nich.

Agregacja

W projekcie należy zastosować relację agregacji.

Agregacja powinna polegać na tym, że jeden obiekt:

- przechowuje kolekcję innych obiektów (np. rekordów danych, źródeł danych lub procesorów),
- nie jest ich jedynym właścicielem,
- może operować na nich jako zbiorze.

Obiekty agregowane muszą móc istnieć niezależnie od obiektu agregującego.

Architektura programu

Projektując rozwiązanie zwróć szczególną uwagę na:

- czytelny podział odpowiedzialności między klasy,
- unikanie nadmiernych zależności,
- możliwość rozbudowy programu bez modyfikowania istniejących klas.

Program powinien być zgodny z zasadą otwarte-zamknięte (Open/Closed Principle).

Wymagania jakościowe

- stosuj adnotacje typów,
- nie używaj zmiennych globalnych,
- oddziel logikę biznesową od I/O,
- przygotuj kod w sposób umożliwiający testowanie jednostkowe,
- dbaj o czytelność i spójność nazewnictwa.

Efekt końcowy

Efektem zadania powinien być jeden program w Pythonie, który po uruchomieniu:

- wczyta dane z pliku wejściowego,

- przetworzy je zgodnie z zaprojektowaną architekturą,
- zapisze wynik do pliku wyjściowego.

Program powinien stanowić spójną całość architektoniczną, a nie zbiór niezależnych przykładów.